ROS Qt Creator Plug-in

Release

Mar 18, 2018

Contents

CHAPTER 1

Installation

1.1 How to Install (Users)

This wiki explains the procedure for installing the ROS Qt Creator Plug-in.

Note: If you primarily want to use this tool for development of other ROS packages (ie: not to work on the plugin itself), please follow the following instructions.

1.1.1 Installation

Important: The install method has changed from using the ppa method to a custom installer. This is to enable the ability to provide richer support leveraging existing ros tools which was not possible using the ppa.

Installation Procedure for Ubuntu 16.04

1. Install Prerequsites:

```
sudo apt install libqtermwidget5-0
```

2. Next proceed to *Qt Installer Procedure*

Important: If previously installed using the ppa please follow the procedure below to remove old version.

```
sudo apt install ppa-purge
sudo ppa-purge -o beineri
sudo ppa-purge levi-armstrong/qt-libraries-xenial
sudo ppa-purge levi-armstrong/ppa
```

Warning: The ppa-purge removes everything installed from the ppa, so if the ppa is used for other development do not purge.

Installation Procedure for Ubuntu 14.04

1. Install Prerequsites:

```
sudo add-apt-repository ppa:levi-armstrong/qt-libraries-trusty
sudo add-apt-repository ppa:levi-armstrong/ppa
sudo apt install libqtermwidget59-0
```

2. Next proceed to *Qt Installer Procedure*

Important: If previously installed using the ppa please follow the procedure below to remove old version.

```
sudo apt install ppa-purge
sudo ppa-purge -o beineri
```

Warning: The ppa-purge removes everything installed from the ppa, so if the ppa is used for other development do not purge.

Archived Versions

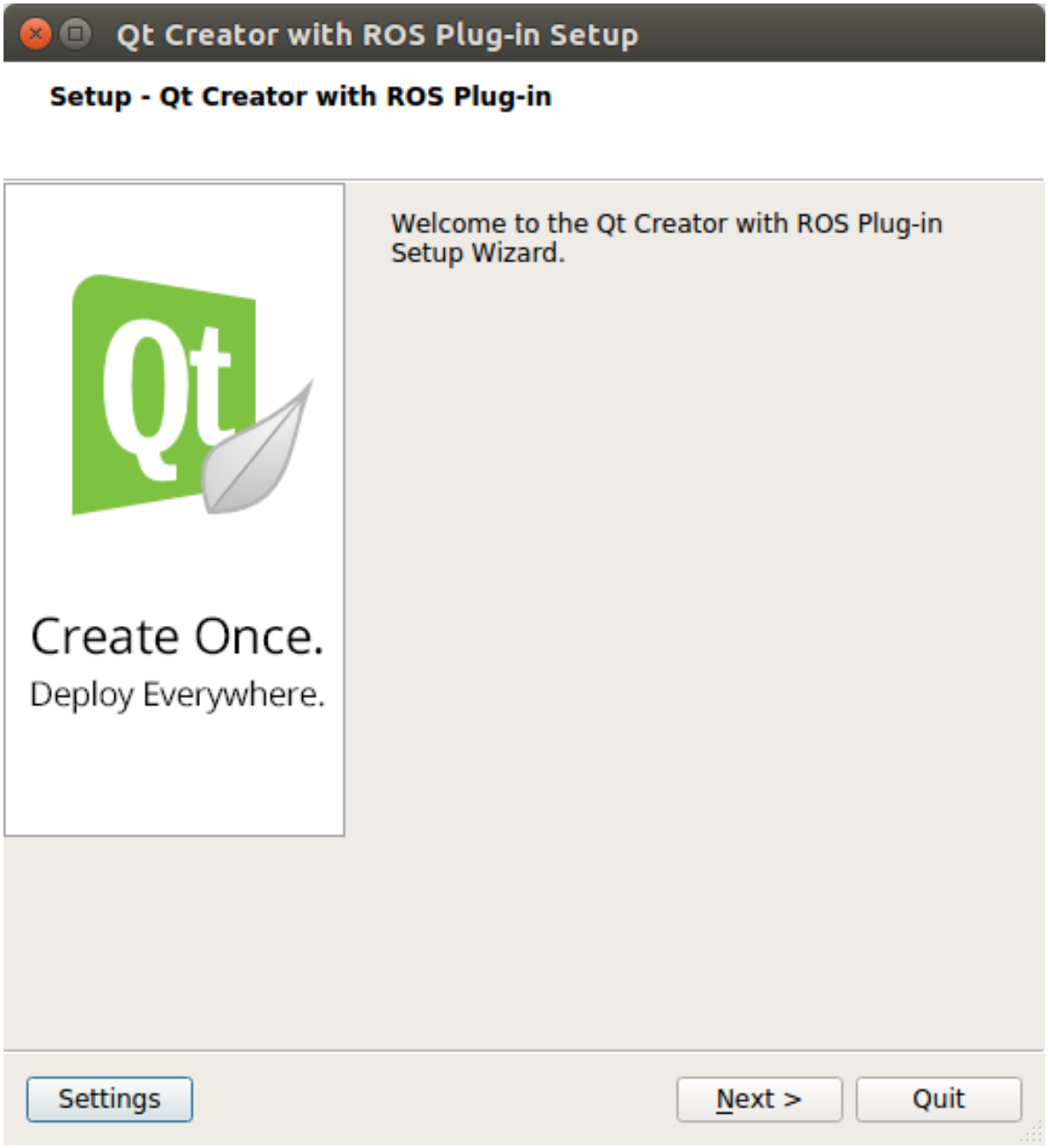
If for some reason you need a version other than the latest, all installers may be found here.

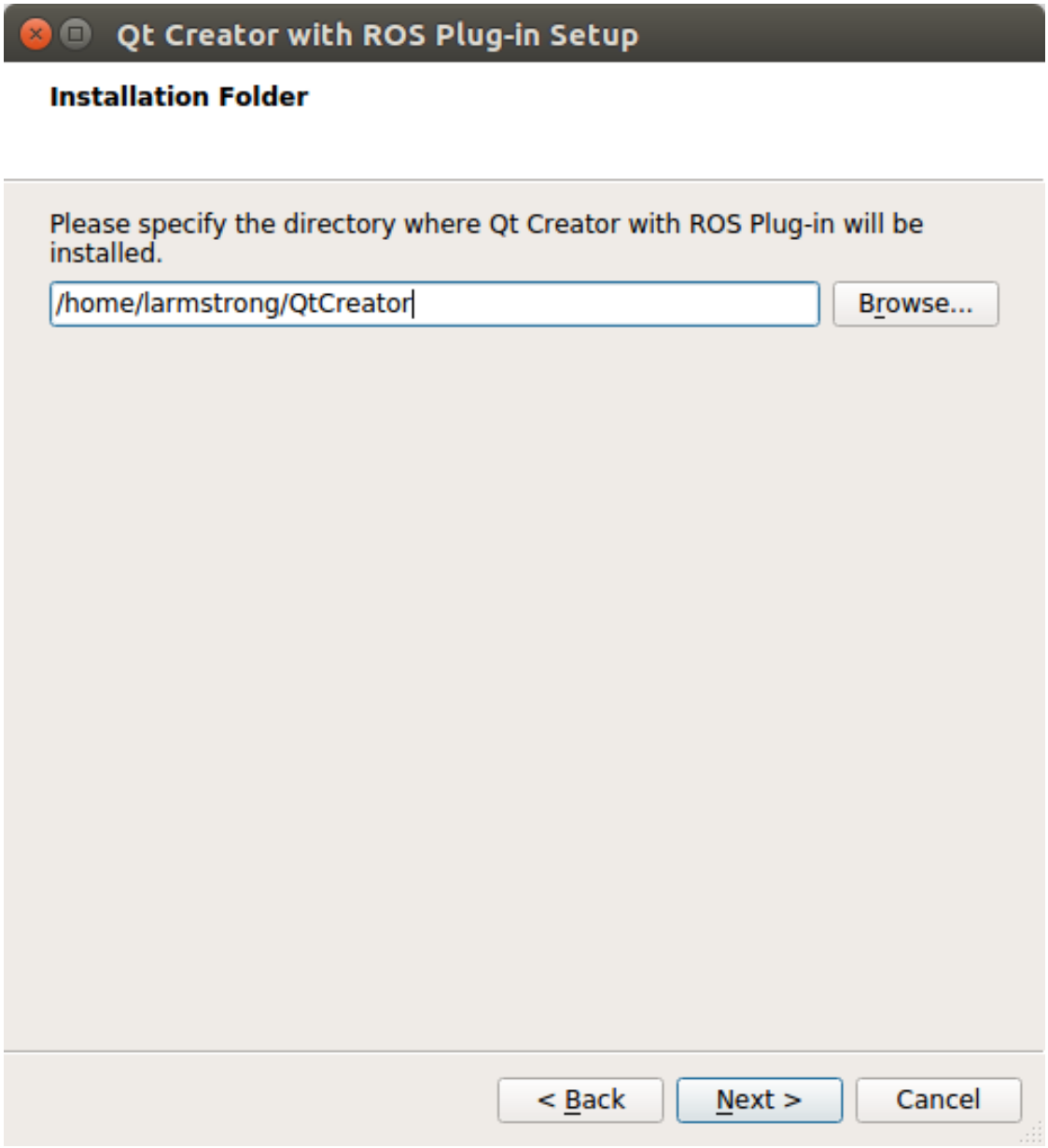
Qt Installer Procedure

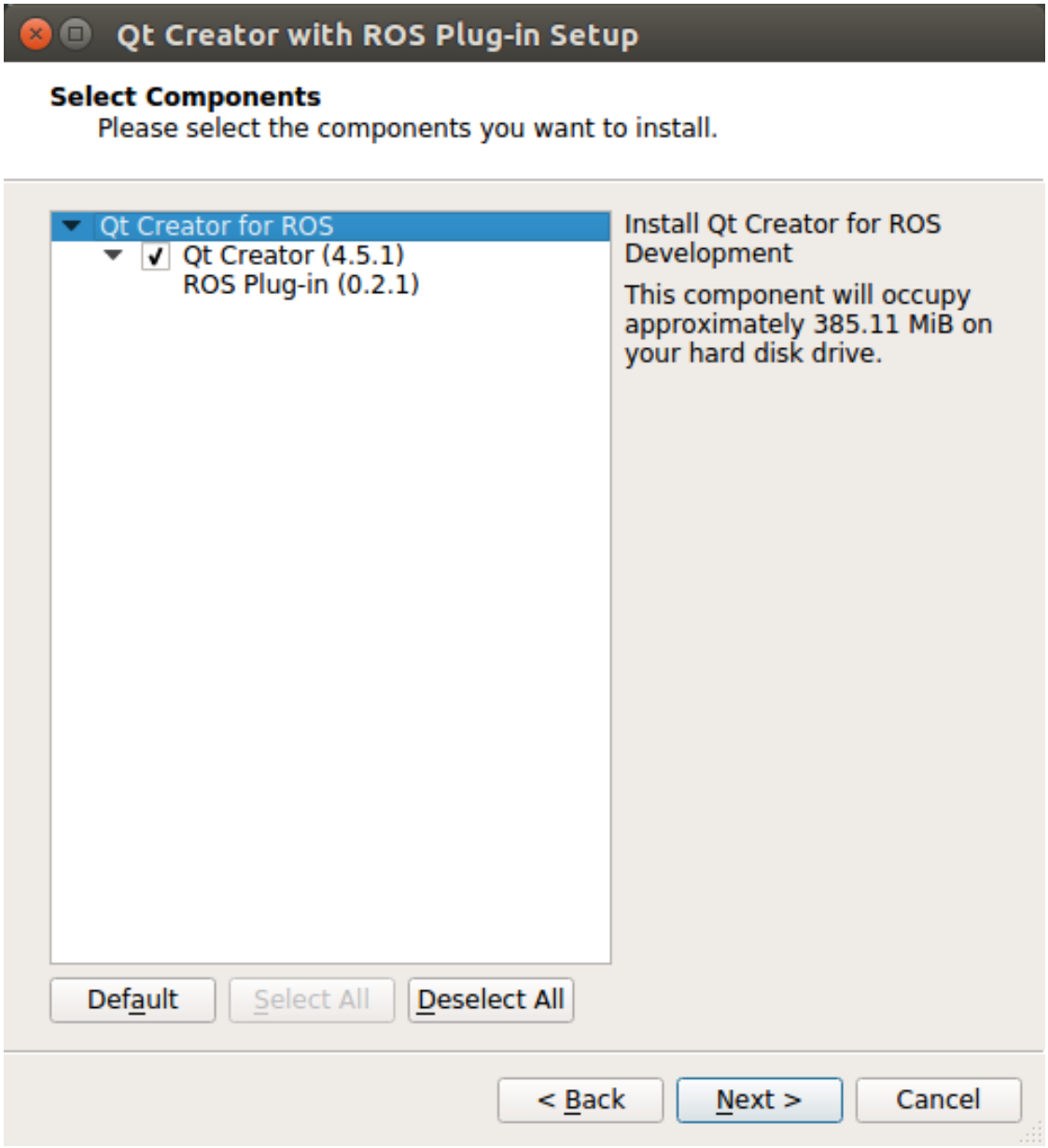
1. Download Installer
   (a) Online Installer (Recommended)
   (b) Offline Installer

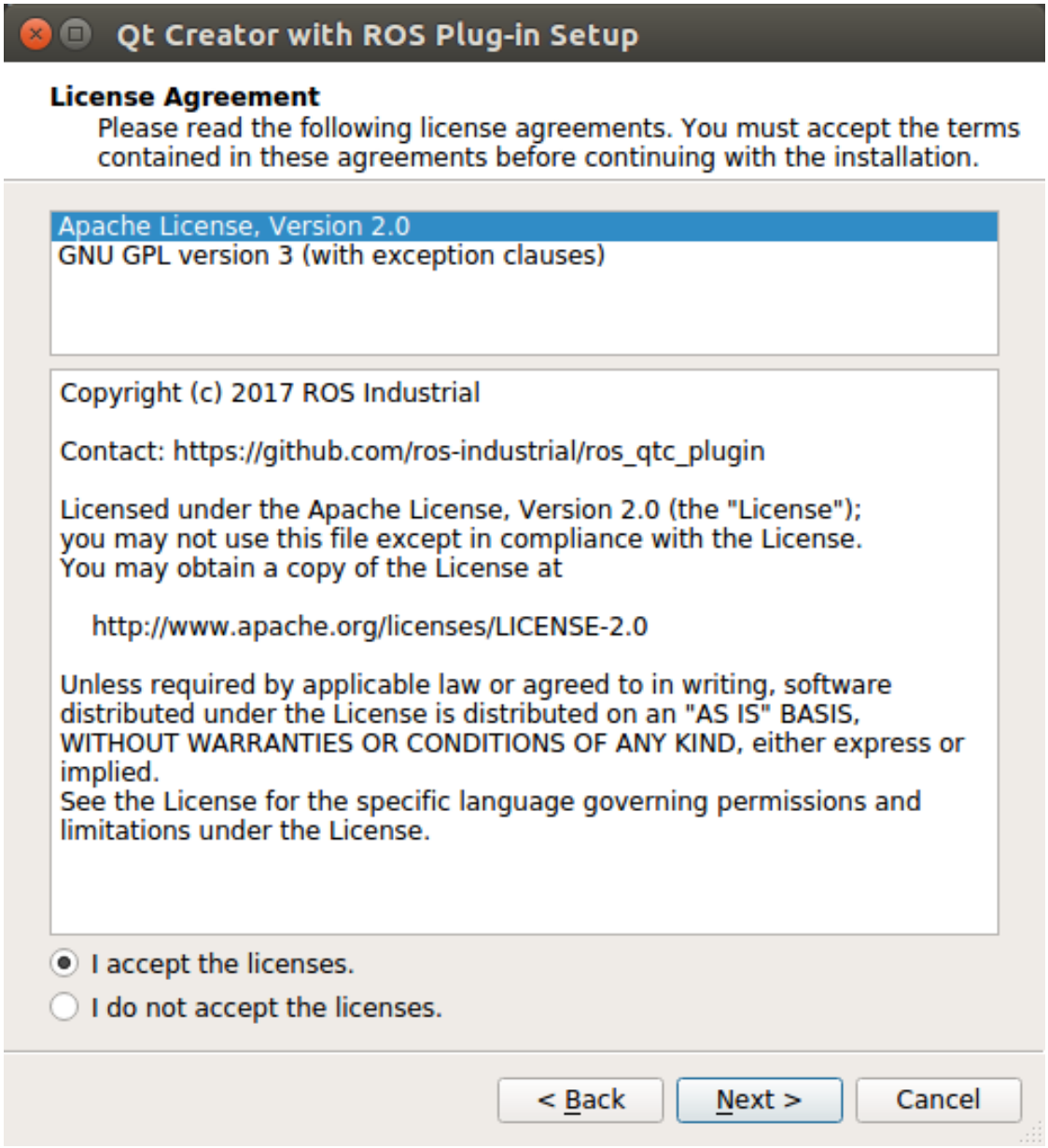
   Note: The Offline Installer is to be used on machines that do not have internet access.

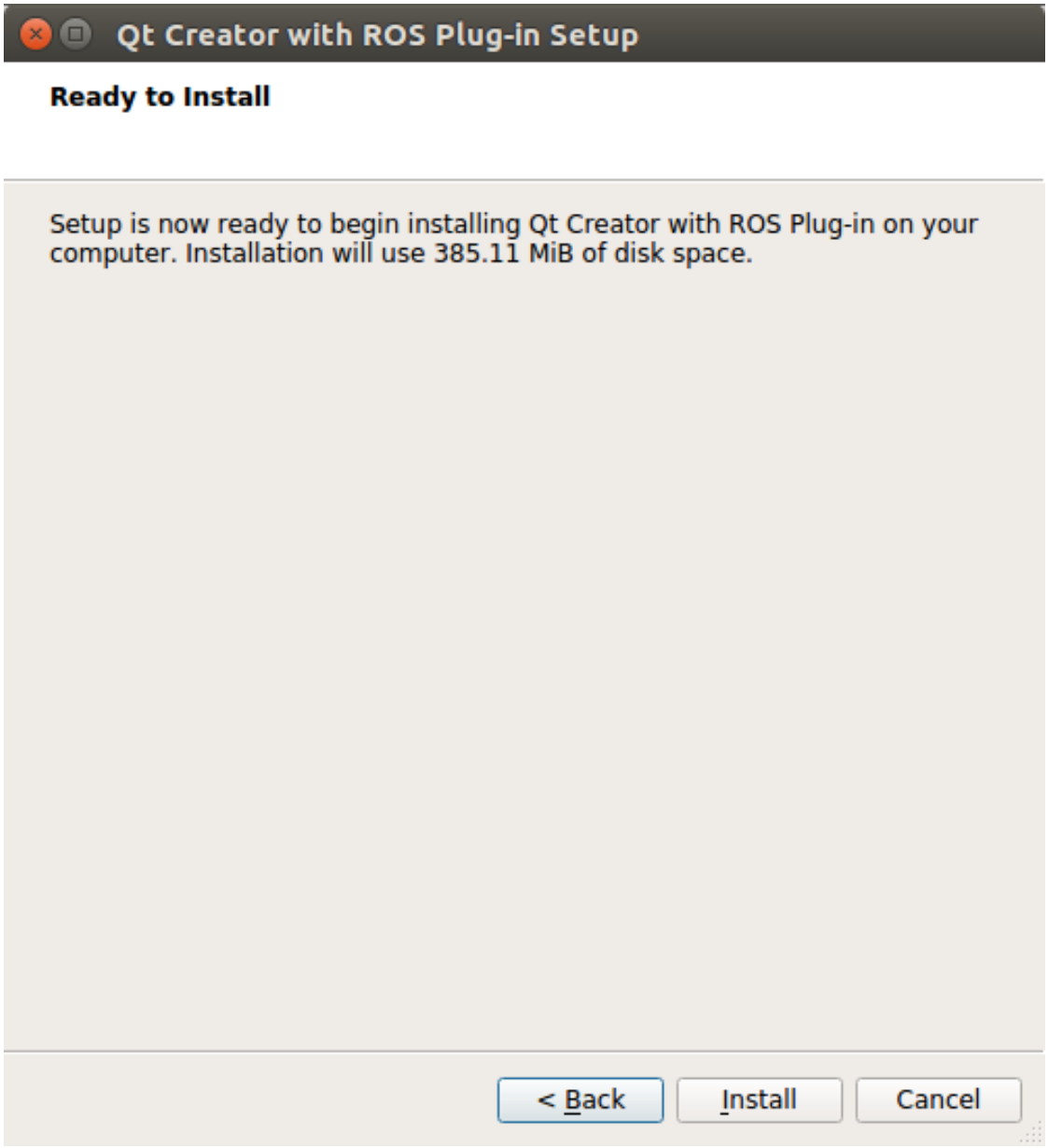
2. Then right click on the installer file, select properties and enable execution under permissions.
3. Next double click the installer and it should open and step throught the installer.

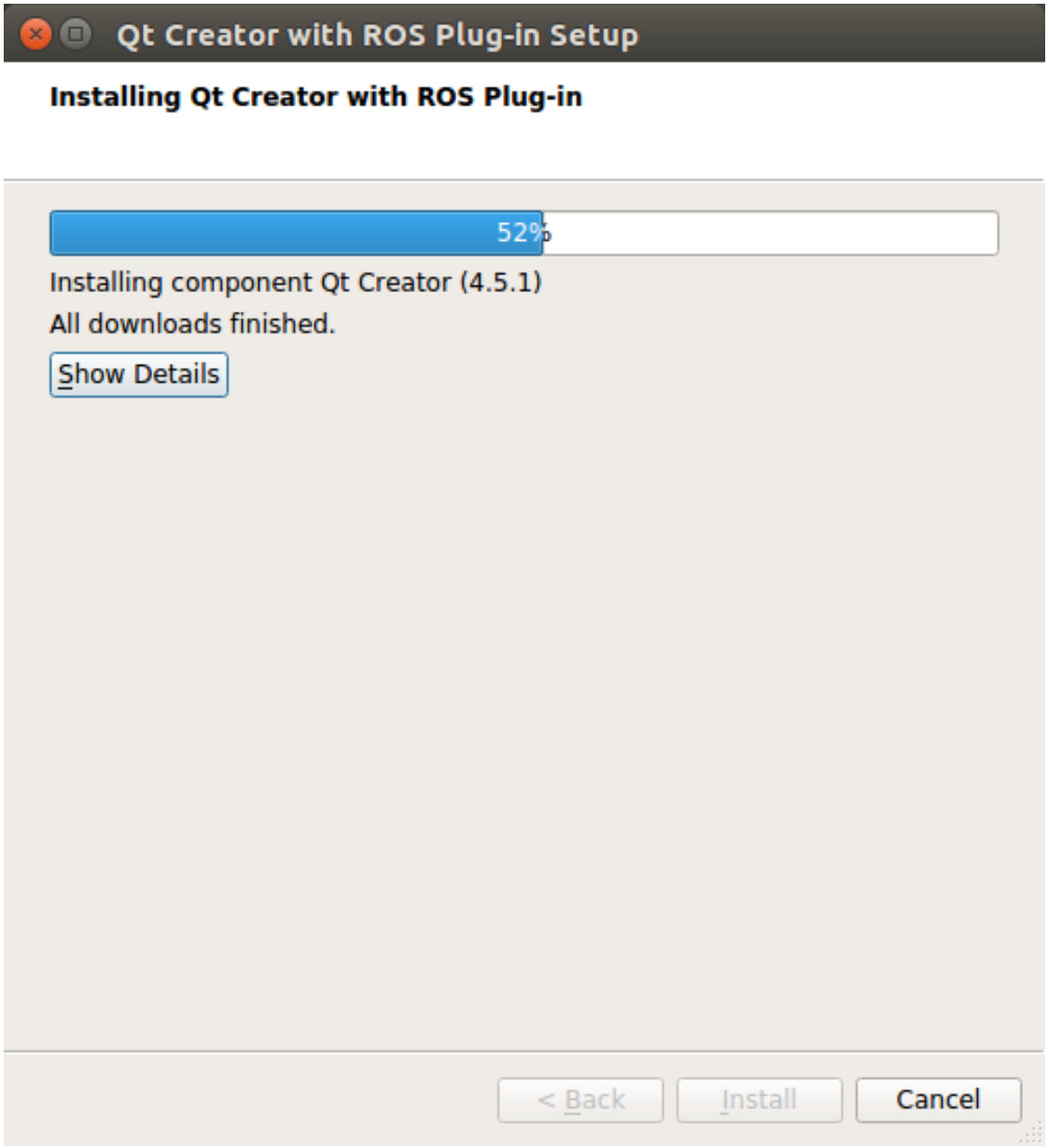

Qt Creator with ROS Plug-in Setup
Setup - Qt Creator with ROS Plug-in
Welcome to the Qt Creator with ROS Plug-in Setup Wizard.
Create Once.
Deploy Everywhere.
Settings
Next >
Quit


Qt Creator with ROS Plug-in Setup
Installation Folder
Please specify the directory where Qt Creator with ROS Plug-in will be installed.
/home/larmstrong/QtCreator
Browse...
< Back
Next >
Cancel

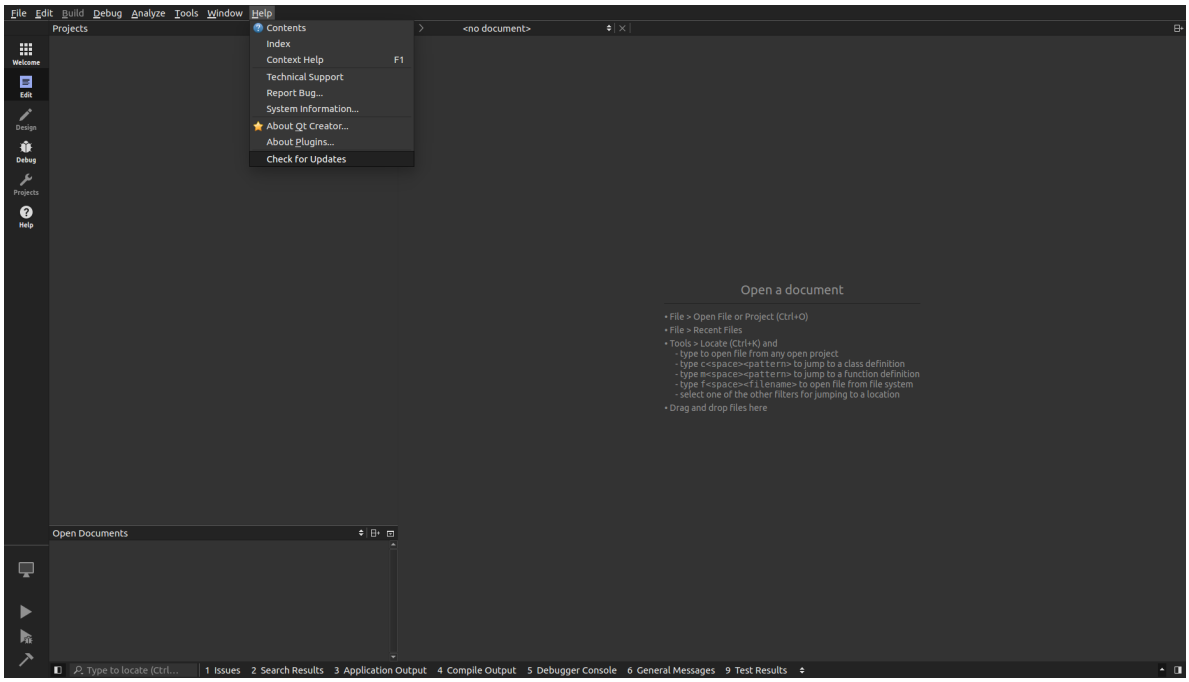

Qt Creator with ROS Plug-in Setup
Select Components
Please select the components you want to install.
Qt Creator for ROS
Qt Creator (4.5.1)
ROS Plug-in (0.2.1)
Install Qt Creator for ROS Development
This component will occupy approximately 385.11 MiB on your hard disk drive.
Default
Select All
Deselect All
< Back
Next >
Cancel


Qt Creator with ROS Plug-in Setup
License Agreement
Please read the following license agreements. You must accept the terms contained in these agreements before continuing with the installation.
Apache License, Version 2.0
GNU GPL version 3 (with exception clauses)
Copyright (c) 2017 ROS Industrial
Contact: https://github.com/ros-industrial/ros_qtc_plugin
Licensed under the Apache License, Version 2.0 (the "License");
you may not use this file except in compliance with the License.
You may obtain a copy of the License at
http://www.apache.org/licenses/LICENSE-2.0
Unless required by applicable law or agreed to in writing, software
distributed under the License is distributed on an "AS IS" BASIS,
WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.
See the License for the specific language governing permissions and
limitations under the License.
I accept the licenses.
I do not accept the licenses.
< Back
Next >
Cancel


Qt Creator with ROS Plug-in Setup
Ready to Install
Setup is now ready to begin installing Qt Creator with ROS Plug-in on your computer. Installation will use 385.11 MiB of disk space.
< Back
Install
Cancel


Qt Creator with ROS Plug-in Setup
Installing Qt Creator with ROS Plug-in
52%
Installing component Qt Creator (4.5.1)
All downloads finished.
Show Details
< Back
Install
Cancel

1. How to get future updates, open Qt Creator and on the menubar under Help select **"Check for Updates"**.



1.1.2 Installation Issues and Conflicts

- No known issues

1.1.3 Testing Plugin.

1. Start Qt Creator
   - Option 1: Launch using the desktop icon.
   - Option 2: Launch from terminal.

   ```
   qtcreator-ros
   ```

2. To verify that the plugin exist, goto File>New File or Project>Projects>Other Project>ROS Workspace. If the ROS Workspace is present then everything built correctly and is ready for development and testing.

1.2 How to Install (Developers)

Note: If you'd like to contribute to the development of the ROS Qt Creator Plug-in, you are considered a *developer*, please follow the following instructions.

1.2.1 Installation

Installation Procedure for Ubuntu 14.04

```
sudo add-apt-repository ppa:levi-armstrong/qt-libraries-trusty
sudo add-apt-repository ppa:levi-armstrong/ppa
sudo apt-get update && sudo apt-get install qt57creator-plugin-ros libqtermwidget57-0-
↪dev
```

Installation Procedure for Ubuntu 16.04

```
sudo add-apt-repository ppa:levi-armstrong/qt-libraries-xenial
sudo add-apt-repository ppa:levi-armstrong/ppa
sudo apt-get update && sudo apt-get install qt57creator-plugin-ros libqtermwidget57-0-
↪dev
```

1.2.2 Configure system to use the new version of Qt

After installation you need to tell qtchooser where to find this install. Replace the two lines in the file below with the location to the local version shown below. Make sure to change *username* and *version* in the file path to match your system.

File:

```
sudo gedit /usr/lib/x86_64-linux-gnu/qt-default/qtchooser/default.conf
```

File content:

```
/opt/qt57/bin
/opt/qt57/lib
```

1.2.3 Run ROS Qt Creator setup script

1. Clone your fork of the repository.

   ```
   git clone -b master https://github.com/<username>/ros_qtc_plugins.git
   ```

2. Next in a terminal, navigate into the repository directory and execute the command below.

   ```
   bash setup.sh -d
   ```

Note: Instruction 2 can can be repeated to get the latest updates for Qt Creator source. The developer must manually update there fork to get the latest version of ros_qtc_plugins.

1.2.4 Testing Plugin

1. Execute the command below or launch using the desktop launcher.

```
qtcreator
```

2. To verify that the plugin exist, goto File>New File or Project>Projects>Other Project>ROS Workspace. If the ROS Workspace is present then everything built correctly and is ready for development and testing.

1.2.5 Debug issues with Plugin

1. Next in a terminal, navigate to the repository ros_qtc_plugin and execute the command below.

```
bash setup.sh -di
```

2. Now launch qtcreator using gdb as shown below and after the plugin segfaults post the trace back in the active/new issue.

```
gdb <local>/qt-creator-build/bin/qtcreator
(gdb) run
```

After error:

```
(gdb) bt
```

CHAPTER 2

Users Help

2.1 Setup Qt Creator for ROS

2.1.1 Setup Ubuntu to allow debugging/ptrace

1. Open file: `sudo gedit /etc/rc.local`
2. Add this line before the exit 0 line: `echo 0 | tee /proc/sys/kernel/yama/ptrace_scope`
3. Reboot computer

2.1.2 Set Theme

If version 3.3.1 or higher was installed you are able to set the theme to dark following the steps bellow.

1. Open Qt Creator
2. Goto: `Tools` > `Options` > `Environment` > `General`
3. There should be a setting for Theme in the "User Interface" group containing a drop down box with two options "default or dark".

2.1.3 Set Syntax Color Schemes

1. Open Qt Creator
2. Goto: `Tools` > `Options` > `Text Editor` > `Font & Colors`
3. There is a drop down box in the "Color Scheme" group where you select different syntax color schemes.
4. Addition schemes can be added as explained in the below links.
   (a) https://github.com/procedural/qtcreator-themes
   (b) https://github.com/alexpana/qt-creator-wombat-theme

2.1.4 Set ROS Code Format

1. Open Qt Creator
2. On the sidebar: `Projects` > `Editor`
   - Changing it globally at `Tools` > `Options` > `C++` or within `Projects` > `Code Style` does not work.

2.1.5 Setup Clang Formatting

1. Install Clang *sudo apt-get install clang-format-3.6*
2. Goto: `Tools` > `Options` > `Environment` > `External Tools`
3. Select: `Add` > `Add Tool`
4. Fill in the information below.
   - Description: Clang Cpp Format
   - Executable: /usr/bin/clang-format-3.6
   - Arguments:

```
-style="{Language: Cpp, AccessModifierOffset: -2, AlignAfterOpenBracket: true,
↪ AlignEscapedNewlinesLeft: false, AlignOperands:   true,␣
↪AlignTrailingComments: true, AllowAllParametersOfDeclarationOnNextLine:␣
↪true, AllowShortBlocksOnASingleLine: false,␣
↪AllowShortCaseLabelsOnASingleLine: false,␣
↪AllowShortIfStatementsOnASingleLine: false, AllowShortLoopsOnASingleLine:␣
↪false, AllowShortFunctionsOnASingleLine: All,␣
↪AlwaysBreakAfterDefinitionReturnType: false,␣
↪AlwaysBreakTemplateDeclarations: false, AlwaysBreakBeforeMultilineStrings:␣
↪false, BreakBeforeBinaryOperators: None, BreakBeforeTernaryOperators: true,␣
↪BreakConstructorInitializersBeforeComma: false, BinPackParameters: true,␣
↪BinPackArguments: true, ColumnLimit:     80,␣
↪ConstructorInitializerAllOnOneLineOrOnePerLine: false,␣
↪ConstructorInitializerIndentWidth: 4, DerivePointerAlignment: false,␣
↪ExperimentalAutoDetectBinPacking: false, IndentCaseLabels: false,␣
↪IndentWrappedFunctionNames: false, IndentFunctionDeclarationAfterType:␣
↪false, MaxEmptyLinesToKeep: 1, KeepEmptyLinesAtTheStartOfBlocks: true,␣
↪NamespaceIndentation: None, ObjCBlockIndentWidth: 2,␣
↪ObjCSpaceAfterProperty: false, ObjCSpaceBeforeProtocolList: true,␣
↪PenaltyBreakBeforeFirstCallParameter: 19, PenaltyBreakComment: 300,␣
↪PenaltyBreakString: 1000, PenaltyBreakFirstLessLess: 120,␣
↪PenaltyExcessCharacter: 1000000, PenaltyReturnTypeOnItsOwnLine: 60,␣
↪PointerAlignment: Left , SpacesBeforeTrailingComments: 1,␣
↪Cpp11BracedListStyle: true, Standard: Cpp11, IndentWidth: 2, TabWidth: 8,␣
↪UseTab: Never, BreakBeforeBraces: Allman, SpacesInParentheses: false,␣
↪SpacesInSquareBrackets: false, SpacesInAngles:  false,␣
↪SpaceInEmptyParentheses: false, SpacesInCStyleCastParentheses: false,␣
↪SpaceAfterCStyleCast: false, SpacesInContainerLiterals: true,␣
↪SpaceBeforeAssignmentOperators: true, ContinuationIndentWidth: 4,␣
↪CommentPragmas:  '^ IWYU pragma:', ForEachMacros:   [ foreach, Q_FOREACH,␣
↪BOOST_FOREACH ], SpaceBeforeParens: ControlStatements, DisableFormat:  ␣
↪false}" -i %{CurrentDocument:FilePath}
```

   - Working directory: %{CurrentProject:Path}
   - Output: Show in Pane

- Error output: Show in Pane
- Environment: No Changes to apply.
- Modifies current document: Checked

5. Select `Apply`
6. Now lets add a quick key.
7. Goto: `Tools` > `Options` > `Environment` > `Keyboard`
8. In the filter box type "Clang" and you should pull up the new tool.
9. In the Shortcut section enter the Target text box and press `Ctrl + Shift + k` to set the shortcut.
10. Now to apply the Clang format to a C++ file open the file in Qt Creator and press `Ctrl + Shift + k` and the file should be formatted correctly.

2.1.6 Preventing Qt Creator form stepping into Boost, Eigen, etc.

1. First clone this repository https://github.com/Levi-Armstrong/gdb-7.7.1.git
2. Follow the instruction in the README file
   (a) ./configure
   (b) make
   (c) sudo checkinstall
3. Goto: `Tools` > `Options` > `Debugger` > `GDB`
4. Add the following code below to the "Additional Startup Commands"

```
skip pending on
python
for root, dirs, files in os.walk("/usr/include/boost/"):
  for file in files:
    if file.endswith(".hpp"):
      cmd = "skip file " + os.path.join(root, file)
      gdb.execute(cmd, True)

for root, dirs, files in os.walk("/usr/include/eigen3/Eigen/"):
  for file in files:
    if file.endswith(".hpp"):
      cmd = "skip file " + os.path.join(root, file)
      gdb.execute(cmd, True)
end
skip enable
```

5. Now when you are stepping through your code it should not step into Boost or Eigen. You can also add additional directories following the same process.
6. Also if you would like to skip a particular function refer to the GDB documentation for instruction. It is something along the lines of *skip function function_name*.

2.2 Debugging Catkin Workspace

2.2.1 Prerequisite

1. Allow ptrace by following these instructions

2.2.2 Attach to a unstarted process

1. Next in Qt Creator browse to the file you wish to debug and insert break points.
2. `Menu Bar` > `Debug` > `Start Debugging` > `Attach to Unstarted Application...`
3. Browse to the executable then select `Start Watching`.
4. Now run your project. `Ctrl` + `R`
5. Now depending on where the breakpoints were placed in qt, it should be stopped at a break point when it reaches one.

2.2.3 Attach to a running process

1. Next in Qt Creator browse to the file you wish to debug and insert break points.
2. Now run your project. `Ctrl` + `R`
3. `Menu Bar` > `Debug` > `Start Debugging` > `Attach to Running Application...`
4. Now select the Process ID and then click the button `Attach to Process`.
5. Now depending on where the breakpoints were placed in qt, it should be stopped at a break point when it reaches one.

Note:

1. Sometime it will be paused at a ros::spin, so check after it has attached and if it is passed and not at a inserted breakpoint click the continue button in the debugger.
2. Some nodes can execute fast enough to where the debugger is not fully attach to the process before it reaches the first breakpoint, resulting in it never stopping at the breakpoint. To solve this add a *sleep(NumberOfSeconds)* command at the start of the node to allow enough time for the debugger to attach. Usually 3 seconds is enough but varies depending on the size of the node.

CHAPTER 3

Developers Help

3.1 Where to find Qt Creator Plug-in Support

There are three reliable ways to get into contact:

1. There is the bug tracker on bugreports.qt.io. People here are very aware of everything entered there, but may not react immediately. But this is the best place for feature requests, bug reports and similar things that should not be forgotten.
2. Our mailing list is actively read by all developers. That is available here which should get most questions answered.
3. The fastest option is IRC: #qt-creator on the freenode network. All developers hang out there, at least while they are in the office. So European business hours tend to work best.

Note: This link provides a good overview of names to ping for specific questions.

CHAPTER 4

Tutorials

1. How to create a catkin workspace.
2. How to configure the run and build settings.